On Solving Large-Scale Finite Minimax Problems using Exponential Smoothing

E. Y. Pee* **and J. O. Royset†**

This paper focuses on finite minimax problems with many functions, and their solutions by means of exponential smoothing. We conduct run-time complexity and rate of convergence analysis of smoothing algorithms and compare them with those of SQP algorithms. We find that smoothing algorithms may have only sublinear rate of convergence, but as shown by our complexity results, their slow rate of convergence may be compensated by small computational work per iteration. We present two smoothing algorithms with active-set strategies that reduce the effect of ill-conditioning using novel precision-parameter adjustment schemes. Numerical results indicate that the proposed algorithms are competitive with other smoothing and SQP algorithms, and they are especially efficient for large-scale minimax problems with a significant number of functions ϵ-active at stationary points.

Key Words. Finite minimax, exponential penalty function, smoothing techniques, active-set strategy.

1 Introduction

There are many applications that can be expressed as finite minimax problems of the form

$$(P) \quad \min_{x \in \mathbb{R}^d} \psi(x), \tag{1}$$

where $\psi : \mathbb{R}^d \to \mathbb{R}$ is defined by

$$\psi(x) \stackrel{\triangle}{=} \max_{j \in Q} f^j(x), \tag{2}$$

and $f^j : \mathbb{R}^d \to \mathbb{R}$, $j \in Q \stackrel{\triangle}{=} \{1, ..., q\}$, $q > 1$, are smooth functions. Minimax problems of the form (P) may occur in engineering design [1], control system design [2], portfolio optimization [3], best polynomial approximation [4], or as subproblems in semi-infinite minimax algorithms [5]. In this paper, we focus on minimax problems with many functions, which may result from finely discretized semi-infinite minimax problems or optimal control problems.

*Graduate Student, Operations Research Department, Naval Postgraduate School, Monterey, California, USA

†Assistant Professor, Operations Research Department, Naval Postgraduate School, Monterey, California, USA

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
2010		**00-00-2010 to 00-00-2010**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
On Solving Large-Scale Finite Minimax Problems using Exponential Smoothing	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Naval Postgraduate School,Operations Research Department,Monterey,CA,93943**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
in review

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**36**	

The non-differentiability of the objective function in (P) poses the main challenge for solving minimax problems, as the usual gradient methods cannot be applied directly. Many algorithms have been proposed to solve (P); see for example [6–8] and references therein. One approach is sequential quadratic programming (SQP), where (P) is first transcribed into the standard nonlinear constrained problem

$$(P') \quad \min_{(x,z)\in\mathbb{R}^{d+1}} \{z \mid f^j(x) - z \leq 0, j \in Q\} \tag{3}$$

and then a SQP algorithm is applied to solve (P'), advantageously exploiting the special structure in the transcribed problem; see [7, 9]. Other approaches also based on (P') include interior point methods [8, 10, 11] and conjugate gradient methods in conjunction with exact penalties and smoothing [12].

Each iteration of the SQP algorithm in [7] solves two quadratic programs (QPs) to compute the main search direction and a modified direction to overcome the Maratos effect. The SQP algorithm in [7] appears especially promising for problems with many sequentially related functions, as in the case of finely discretized semi-infinite minimax problems, due to its aggressive active-set strategy. Recently, a SQP algorithm was proposed in [9], where the modified direction is obtained by solving a system of linear equations. This reduces the number of QPs from two to one per iteration, while still retaining global convergence as well as superlinear rate of convergence. There is no active-set strategy in [9].

In general, an active-set strategy only considers functions that are active or almost active (ϵ-active) at the current iterate, and thus greatly reduces the number of function and gradient evaluations at each iteration of an algorithm. While the number of iterations to solve a problem to required precision may increase, the overall effect may be a significant reduction in the total number of function and gradient evaluations of the algorithm. The numerical results for an active-set minimax algorithm in [13] give a 75% reduction in the number of gradient evaluations, when compared against the same algorithm without the active-set strategy. Significant reduction in computing time is also reported for active-set strategies in [7].

In smoothing algorithms, see for example [6, 12–15], the exponential penalty function introduced in [16] is used to produce a smooth (twice continuously differentiable) function that approximates $\psi(\cdot)$. Since the problem remains unconstrained, one can use any standard unconstrained optimization algorithm to solve the smoothed problem such as the Armijo

Gradient or Newton methods [6] and Quasi-Newton method [13].

A fundamental challenge of smoothing algorithms is that the smoothed problem becomes increasingly ill-conditioned as the approximation gets more accurate. Hence, an unconstrained optimization solver may experience numerical difficulties and slow convergence. Consequently, the use of smoothing techniques is complicated by the need to balance accuracy of approximation and problem ill-conditioning. An attempt to address these shortcomings was first made in [15], where a precision parameter for the smooth approximation is initially set to a pre-selected value and is then increased by a fixed factor (specifically 2) at each consecutive iteration. Effectively, the algorithm is solving a sequence of gradually more accurate approximations. However, the main problem with this open-loop scheme is its sensitivity to the selection of the multiplication factor, as can be seen from the numerical results in [6].

In [6], the authors propose an adaptive precision-parameter adjustment scheme with exponential smoothing to ensure that the precision parameter is kept small (and thus controlling the ill-conditioning) when far from a stationary solution, and is increased as a stationary solution is approached. The authors use the norm of the smoothed function gradient as a proxy for the distance to a stationary solution. When the gradient norm of the smoothed function falls below a user-specified threshold, the precision parameter is increased to a level that ensures that the gradient norm falls within two user-specified bounds. The numerical results show that this adaptive scheme produces a much better management of ill-conditioning than with open-loop schemes. The smoothing algorithms in [15] and [6] do not incorporate any active-set strategy.

Using the same adaptive precision-parameter adjustment scheme as in [6], the authors in [13] present a new active-set strategy that can be used in conjunction with exponential smoothing for specifically tackling large-scale (large q) minimax problems. We note that the convergence result in Theorem 3.3 of [13] may be slightly incorrect as it claims stationarity for all accumulation points of a sequence constructed by their algorithm. However, their proof relies on [6], which guarantees stationarity for only one accumulation point.

While the literature describes several smoothing algorithms for (P), there appears to be no run-time complexity and rate of convergence analysis of such algorithms. Moreover, we find no comprehensive empirical comparison of run times for SQP and smoothing algorithms. In this paper, we present run-time complexity and rate of convergence results for smoothing

algorithms and compare them with those of SQP algorithms. We propose two new active-set smoothing algorithms based on [6, 13] and present computational test results for large-scale problem instances.

The next section describes the exponential smoothing technique and its properties. Section 3 defines a smoothing algorithm and discusses run-time complexity and rate of convergence. Section 4 presents two new smoothing algorithms and their proofs of convergence. Section 5 contains numerical test results.

2 Exponential Smoothing

In this section, we describe the exponential smoothing technique, include for completeness some known results, and show that the technique leads to consistent approximations (see Section 3.3 of [17]).

For ease of analysis of active-set strategies, we consider the problem

$$(P_\Omega) \quad \min_{x\in\mathbb{R}^d} \psi_\Omega(x), \tag{4}$$

where

$$\psi_\Omega(x) \stackrel{\triangle}{=} \max_{j\in\Omega} f^j(x), \tag{5}$$

and $\Omega \subset Q$. When $\Omega = Q$, (P_Q) is identical to (P). Next, for any $p > 0$ and $\Omega \subset Q$, we consider a smooth approximating problem to (P_Ω), called the smoothed problem,

$$(P_{p\Omega}) \quad \min_{x\in\mathbb{R}^d} \psi_{p\Omega}(x), \tag{6}$$

where

$$\psi_{p\Omega}(x) \stackrel{\triangle}{=} \frac{1}{p}\log\left(\sum_{j\in\Omega}\exp\left(pf^j(x)\right)\right) \tag{7}$$

$$= \psi_\Omega(x) + \frac{1}{p}\log\left(\sum_{j\in\Omega}\exp\left(p(f^j(x)-\psi_\Omega(x))\right)\right) \tag{8}$$

is the exponential penalty function, with $\log(\cdot)$ denoting the natural logarithm. This smoothing technique was first introduced in [16] and later used in [6, 12–15].

We denote the set of active functions at $x \in \mathbb{R}^d$ by $\widehat{\Omega}(x) \stackrel{\triangle}{=} \{j \in \Omega | f^j(x) = \psi_\Omega(x)\}$. Except as specifically stated in Appendix A, we denote components of a vector by superscripts. We also let $\mathbb{N}$ denote the set of positive integers and $\mathbb{N}_0 \stackrel{\triangle}{=} \mathbb{N} \cup \{0\}$.

The parameter $p > 0$ is the smoothing precision parameter, where a larger p implies higher precision as formalized by the following proposition; see for example [13].

Proposition 2.1. *Suppose that $\Omega \subset Q$ and $p > 0$.*

(i) If the functions $f^j(\cdot)$, $j \in \Omega$, are continuous, then $\psi_{p\Omega}(\cdot)$ is continuous and decreases monotonically as p increases.

(ii) For any $x \in \mathbb{R}^d$,

$$0 \leq \frac{\log|\widehat{\Omega}(x)|}{p} \leq \psi_{p\Omega}(x) - \psi_{\Omega}(x) \leq \frac{\log|\Omega|}{p}, \tag{9}$$

where $|\cdot|$ represents the cardinality operator.

(iii) If the functions $f^j(\cdot)$, $j \in \Omega$, are continuously differentiable, then $\psi_{p\Omega}(\cdot)$ is continuously differentiable, with gradient

$$\nabla\psi_{p\Omega}(x) = \sum_{j\in\Omega} \mu_p^j(x)\nabla f^j(x), \tag{10}$$

where

$$\mu_p^j(x) \stackrel{\triangle}{=} \frac{\exp(pf^j(x))}{\sum_{k\in\Omega}\exp(pf^k(x))} = \frac{\exp(p[f^j(x)-\psi_{\Omega}(x)])}{\sum_{k\in\Omega}\exp(p[f^k(x)-\psi_{\Omega}(x)])} \in (0,1), \tag{11}$$

and $\sum_{j\in\Omega}\mu_p^j(x) = 1$.

(iv) If the functions $f^j(\cdot)$, $j \in \Omega$, are twice continuously differentiable, then $\psi_{p\Omega}(\cdot)$ is twice continuously differentiable, with Hessian

$$\begin{aligned}\nabla^2\psi_{p\Omega}(x) &= \sum_{j\in\Omega}\mu_p^j(x)\nabla^2 f^j(x) + p\sum_{j\in\Omega}\mu_p^j(x)\nabla f^j(x)\nabla f^j(x)^T \\ &\quad -p\left[\sum_{j\in\Omega}\mu_p^j(x)\nabla f^j(x)\right]\left[\sum_{j\in\Omega}\mu_p^j(x)\nabla f^j(x)\right]^T. \end{aligned} \tag{12}$$

□

Assumption 2.1. *We assume that the functions $f^j(\cdot), j \in Q$, are twice continuously differentiable.* □

The next lemma can be deduced from Lemma 2.2 of [6].

Lemma 2.1. *Suppose that Assumption 2.1 holds. Then, for every bounded set $S \subset \mathbb{R}^d$, there exists an $L < \infty$ such that*

$$\langle y, \nabla^2 \psi_{p\Omega}(x) y \rangle \leq pL\|y\|^2, \tag{13}$$

for all $x \in S, y \in \mathbb{R}^d$, $\Omega \subset Q$, and $p \geq 1$. □

A continuous, nonpositive optimality function for (P_Ω) is given by

$$\theta_\Omega(x) \stackrel{\triangle}{=} -\min_{\mu \in \Sigma_\Omega} \left\{ \sum_{j \in \Omega} \mu^j (\psi_\Omega(x) - f^j(x)) + \tfrac{1}{2} \left\| \sum_{j \in \Omega} \mu^j \nabla f^j(x) \right\|^2 \right\}, \tag{14}$$

where

$$\Sigma_\Omega \stackrel{\triangle}{=} \left\{ \mu \in \mathbb{R}^{|\Omega|} | \mu^j \geq 0 \text{ for all } j \in \Omega, \sum_{j \in \Omega} \mu^j = 1 \right\}, \tag{15}$$

which results in the following optimality condition for (P_Ω); see Theorems 2.1.1, 2.1.3, and 2.1.6 of [17].

Proposition 2.2. *Suppose that Assumption 2.1 holds and that $\Omega \subset Q$. If $x^* \in \mathbb{R}^d$ is a local minimizer for (P_Ω), then $\theta_\Omega(x^*) = 0$.* □

The continuous, nonpositive optimality function

$$\theta_{p\Omega}(x) \stackrel{\triangle}{=} -\tfrac{1}{2}\|\nabla \psi_{p\Omega}(x)\|^2 \tag{16}$$

characterizes stationary points of $(P_{p\Omega})$ as stated in the next proposition; see Proposition 1.1.6 in [17].

Proposition 2.3. *Suppose that Assumption 2.1 holds, $p > 0$, and $\Omega \subset Q$. If $x^* \in \mathbb{R}^d$ is a local minimizer for $(P_{p\Omega})$, then $\theta_{p\Omega}(x^*) = 0$.* □

We next show that the exponential smoothing technique leads to consistent approximations (see Section 3.3 in [17]), which ensures that globally and locally optimal points as well as stationary points of $(P_{p\Omega})$ converge to corresponding points of (P_Ω), as $p \to \infty$. Consistent approximations also facilitate the construction of implementable algorithms for (P); see Algorithm 4.1 below.

We define *consistent approximation* as on page 399 of [17].

Definition 2.1. *For any $\Omega \subset Q$, $p > 0$, we say that the pair $((P_{p\Omega}), \theta_{p\Omega}(\cdot))$ is a consistent approximation to $((P_\Omega), \theta_\Omega(\cdot))$ if (i) $(P_{p\Omega})$ epi-converges to (P_Ω), as $p \to \infty$, and (ii) for any sequences $\{x_i\}_{i=0}^\infty \subset \mathbb{R}^d$ and $\{p_i\}_{i=0}^\infty, p_i > 0$ for all i, and $x^* \in \mathbb{R}^d$ such that $x_i \to x^*$ and $p_i \to \infty$, as $i \to \infty$, $\limsup_{i\to\infty} \theta_{p_i\Omega}(x_i) \leq \theta_\Omega(x^*)$.* □

Theorem 2.1. *Suppose that Assumption 2.1 holds, $p > 0$, and $\Omega \subset Q$. Then, the pair $((P_{p\Omega}), \theta_{p\Omega}(\cdot))$ is a consistent approximation to $((P_\Omega), \theta_\Omega(\cdot))$.*

Proof. We follow the proofs of Lemmas 4.3 and 4.4 in [18], but simplify the arguments as [18] deals with min-max-min problems. By Theorem 3.3.2 of [17], Proposition 2.1(ii), and the continuity of $\psi_\Omega(\cdot)$, it follows that $(P_{p\Omega})$ epi-converges to (P_Ω), as $p \to \infty$.

We next consider the optimality functions. Let $\{x_i\}_{i=0}^\infty \subset \mathbb{R}^d$ and $\{p_i\}_{i=0}^\infty, p_i > 0$ for all i, be arbitrary sequences and $x^* \in \mathbb{R}^d$ be such that $x_i \to x^*$ and $p_i \to \infty$, as $i \to \infty$. Since $\mu_p^j(x) \in (0,1)$ for any $j \in \Omega$, $p > 0$, and $x \in \mathbb{R}^d$, $\{\mu_{p_i}(x_i)\}_{i=0}^\infty$ is a bounded sequence in $\mathbb{R}^{|\Omega|}$ with at least one convergent subsequence. Hence, for every such subsequence $K \subset \mathbb{N}_0$, there exists a $\mu_\infty \in \Sigma_\Omega$ such that $\mu_{p_i}(x_i) \to^K \mu_\infty$, as $i \to \infty$. Moreover, since $\mu_\infty \in \Sigma_\Omega$, $\sum_{j\in\Omega} \mu_\infty^j = 1$.

If $j \notin \hat{\Omega}(x^*)$, then there exist a $t > 0$ and $i_0 \in \mathbb{N}$ such that $f^j(x_i) - \psi_\Omega(x_i) \leq -t$ for all $i \geq i_0$. Hence, from (11), $\mu_{p_i}^j(x_i) \to 0$, as $i \to \infty$, and therefore $\mu_\infty^j = 0$. By continuity of $\nabla f^j(\cdot)$, $j \in \Omega$,

$$\theta_{p_i\Omega}(x_i) \to^K -\frac{1}{2}\|\sum_{j\in\Omega} \mu_\infty^j \nabla f^j(x^*)\|^2 \triangleq \theta_{\infty\Omega}(x^*), \tag{17}$$

as $i \to \infty$. Since $\mu_\infty \in \Sigma_\Omega$ and $\mu_\infty^j = 0$ for all $j \notin \hat{\Omega}(x^*)$, we find in view of (14) that

$$\theta_{\infty\Omega}(x^*) \;=\; -\sum_{j\in\Omega} \mu_\infty^j(\psi_\Omega(x^*) - f^j(x^*)) - \frac{1}{2}\|\sum_{j\in\Omega} \mu_\infty^j \nabla f^j(x^*)\|^2 \leq \theta_\Omega(x^*). \tag{18}$$

This completes the proof. □

3 Run-Time Complexity and Rate of Convergence

In this section, we focus on the run-time complexity and rate of convergence of smoothing algorithms. Specifically, we deal with the following simple smoothing algorithm for solving (P) based on application of the Armijo Gradient Method[1] to (P_{pQ}).

[1]The Armijo Gradient Method uses the steepest descent search direction and the Armijo stepsize rule to solve an unconstrained problem; see for example Algorithm 1.3.3 of [17].

Algorithm 3.1. Smoothing Armijo Gradient Algorithm

Data: $t > 0, x_0 \in \mathbb{R}^d$.

Parameter: $\delta \in (0, 1)$.

Step 1. Set $p^* = \log q/((1-\delta)t)$.

Step 2. Generate a sequence $\{x_i\}_{i=0}^{\infty}$ by applying Armijo Gradient Method to (P_{p^*Q}). □

We denote the optimal value of (P) (when it exists) by ψ^*, the optimal value of (P_{pQ}) (when it exists) by ψ^*_{pQ} for any $p > 0$, and the optimal solution of (P_{pQ}) (when it exists) by x^*_{pQ}. Algorithm 3.1 has the following property.

Proposition 3.1. *Suppose that Step 2 of Algorithm 3.1 has generated a point $x_i \in \mathbb{R}^d$ such that $\psi_{p^*Q}(x_i) - \psi^*_{p^*Q} \leq \delta t$. Then, $\psi(x_i) - \psi^* \leq t$.*

Proof. The result follows directly from (9) and the selection of p^*. □

Proposition 3.1 shows that we can obtain a near-optimal solution of (P) by approximately solving (P_{pQ}) for a sufficiently large p. As discussed in Section 1, Algorithm 3.1 will be prone to ill-conditioning. Adaptive schemes for adjusting the precision parameter p and the use of another method in Step 2 may perform better in practice. However, the following study of run-time complexity and rate of convergence of Algorithm 3.1 provides fundamental insights into smoothing algorithms in general.

We start with some intermediate results that utilize the following convexity assumption.

Assumption 3.1. *Suppose that $f^j(\cdot), j \in Q$, are twice continuously differentiable and there exist $0 < m \leq M < \infty$ such that*

$$m\|y\|^2 \leq \langle y, \nabla^2 f^j(x) y\rangle \leq M\|y\|^2, \tag{19}$$

for all $x, y \in \mathbb{R}^d$, and for all $j \in Q$. □

Lemma 3.1. *Suppose that Assumption 3.1 holds. For any $x, y \in \mathbb{R}^d$ and $p > 0$,*

$$m\|y\|^2 \leq \left\langle y, \nabla^2 \psi_{pQ}(x) y\right\rangle. \tag{20}$$

Proof. From (12) and (19), we obtain that

$$\begin{aligned}
\left\langle y, \nabla^2 \psi_{pQ}(x) y \right\rangle &= \sum_{j\in Q} \mu_p^j(x) \left\langle y, \nabla^2 f^j(x) y \right\rangle + p \sum_{j\in Q} \mu_p^j(x) \left\langle y, \nabla f^j(x) \nabla f^j(x)^T y \right\rangle \\
&- p \left\langle y, \left[\sum_{j\in Q} \mu_p^j(x) \nabla f^j(x)\right] \left[\sum_{j\in Q} \mu_p^j(x) \nabla f^j(x)\right]^T y \right\rangle \\
&= \sum_{j\in Q} \mu_p^j(x) \left\langle y, \nabla^2 f^j(x) y \right\rangle + p \sum_{j\in Q} \mu_p^j(x) \left\langle y, \nabla f^j(x) \right\rangle^2 \\
&- p \left\langle y, \left[\sum_{j\in Q} \mu_p^j(x) \nabla f^j(x)\right] \right\rangle^2 \\
&\geq m\|y\|^2 + p \sum_{j\in Q} \mu_p^j(x) \left\langle y, \nabla f^j(x) \right\rangle^2 - p \left\langle y, \left[\sum_{j\in Q} \mu_p^j(x) \nabla f^j(x)\right] \right\rangle^2 .
\end{aligned}$$

Hence, we only need to show that the difference of the last two terms is nonnegative. Let $g : \mathbb{R}^d \to \mathbb{R}$ be defined as $g(z) = \langle y, z\rangle^2$. The function g is a composition of a convex function with a linear function, so it is convex; see for example Proposition 2.1.5 of [19]. Hence, it follows from Jensen's inequality (see for example page 6 of [19]) that

$$\sum_{j\in Q} \mu_p^j(x) g\left(\nabla f^j(x)\right) \geq g\left(\sum_{j\in Q} \mu_p^j(x) \nabla f^j(x)\right). \tag{21}$$

Since $p > 0$, the result follows. □

Proposition 3.2. *Suppose that Assumption 3.1 holds and $p \geq 1$. Then, the rate of convergence for the Armijo Gradient Method to solve (P_{pQ}) is linear with coefficient $1 - k/p$, for some $k \in (0, 1)$. That is, for any sequence $\{x_i\}_{i=0}^{\infty} \subset \mathbb{R}^d$ generated by the Armijo Gradient Method when applied to (P_{pQ}),*

$$\psi_{pQ}(x_{i+1}) - \psi_{pQ}^* \leq \left(1 - \frac{k}{p}\right) [\psi_{pQ}(x_i) - \psi_{pQ}^*)] \quad \text{for all } i \in \mathbb{N}_0. \tag{22}$$

Proof. Based on Lemmas 2.1 and 3.1, for every bounded set $S \subset \mathbb{R}^d$ there exist an $L \in [m, \infty)$ such that

$$m\|y\|^2 \leq \left\langle y, \nabla^2 \psi_{pQ}(x) y \right\rangle \leq pL\|y\|^2, \tag{23}$$

for all $x \in S$ and $y \in \mathbb{R}^d$ and $p \geq 1$. Hence, we deduce from Theorem 1.3.7 of [17] that the rate of convergence for Armijo Gradient Method to solve (P_{pQ}) is

$$1 - \frac{4m\beta\alpha(1-\alpha)}{pL} \in (0, 1), \tag{24}$$

where $\alpha, \beta \in (0,1)$ are the Armijo line search parameters. Hence, $k = 4m\beta\alpha(1-\alpha)/L$, which is less than unity because $\alpha(1-\alpha) \in (0, 1/4]$. □

In order to analyze the run-time complexity of Algorithm 3.1, we need an assumption on the complexity of function and gradient evaluations.

Assumption 3.2. *We assume that there exist constants $c, c' < \infty$ such that for any $d \in \mathbb{N}$, $j \in Q$, and $x \in \mathbb{R}^d$, the computational work to evaluate $f^j(x)$ and $\nabla f^j(x)$ is no larger than cd and $c'd^2$, respectively.* □

Assumption 3.2 holds for all problem instances considered in this paper (see Appendix A) and appears reasonable for many practical situations. The following result can easily be modified to account for other assumption about work per function and gradient evaluation.

Theorem 3.1. *Suppose that Assumptions 3.1 and 3.2 hold. For any tolerance $t \in (0, \log q)$, there exists a constant $c_t < \infty$ such that the computational work in Algorithm 3.1 to generate $\{x_i\}_{i=0}^n$, with the last iterate satisfying $\psi(x_n) - \psi^* \leq t$, is no larger than $c_t q d^2 \log q$.*

Proof. Since $p^* = \log q/((1-\delta)t) > 1$ for $t \in (0, \log q)$, Proposition 3.2 applies and we find that the number of iterations of the Armijo Gradient Method to obtain $\{x_i\}_{i=0}^n$ such that $\psi_{p^*Q}(x_n) - \psi^*_{p^*Q} \leq \delta t$ is no larger than

$$\left\lceil \frac{\log \frac{\delta t}{t_0}}{\log(1 - \frac{k}{p^*})} \right\rceil, \tag{25}$$

where k is as in Proposition 3.2, $t_0 = \psi(x_0) - \psi^*$, and $\lceil \cdot \rceil$ denotes the ceiling operator. Since the main computational work in each iteration for the Armijo Gradient Method is to determine $\nabla \psi_{p^*Q}(x_i)$, see (10), it follows by Assumption 3.2 that there exists a $c^* < \infty$ such that the computational work in each iteration of the Armijo Gradient Method when applied to (P_{p^*Q}) is no larger than $c^* q d^2$. Hence, the computational work in Algorithm 3.1 to generate $\{x_i\}_{i=0}^n$, with $\psi_{p^*Q}(x_n) - \psi^*_{p^*Q} \leq \delta t$, is no larger than

$$c^* q d^2 \left\lceil \frac{\log \frac{\delta t}{t_0}}{\log(1 - \frac{k}{p^*})} \right\rceil. \tag{26}$$

Since $p^* = \log q/((1-\delta)t)$, it follows from Proposition 3.1 that the computational work in Algorithm 3.1 to generate $\{x_i\}_{i=0}^n$, with $\psi(x_n) - \psi^* \leq t$, is no larger than

$$c^* q d^2 \left\lceil \frac{\log \frac{\delta t}{t_0}}{\log\left(1 - \frac{k(1-\delta)t}{\log q}\right)} \right\rceil \leq c^* q d^2 \left\lceil \frac{\log \frac{\delta t}{t_0}}{-\frac{k(1-\delta)t}{\log q}} \right\rceil, \tag{27}$$

where we use the fact that $|\log x| \geq |x-1|$ for $x \in (0,1]$. The result then follows. □

Focusing on q, we see from Theorem 3.1 and its proof that the number of iterations for Algorithm 3.1 to achieve a near-optimal solution of (P) is $O(\log q)$. Moreover, the run-time complexity of Algorithm 3.1 to achieve a near-optimal solution of (P) is $O(q \log q)$.

For comparison, we next consider the run-time complexity of a SQP algorithm to achieve a near-optimal solution of (P). The main computational work in an iteration of a SQP algorithm involve solving a convex QP with $d+1$ variables and q inequality constraints [7]. Introducing slack variables to convert into standard form, this subproblem becomes a convex QP with $d+1+q$ variables and q equality constraints. Based on [20], the number of operations to solve the converted QP is $O((d+1+q)^3)$. Assuming that the number of iterations a SQP algorithm needs to achieve a near-optimal solution of (P) is $O(1)$, and again focusing on q, the run-time complexity of a SQP algorithm to achieve a near-optimal solution of (P) is no better than $O(q^3)$. This complexity, when compared with $O(q \log q)$ of Algorithm 3.1, indicates that smoothing algorithms may be more efficient than SQP algorithms for minimax problems with many functions.

Next, we consider the rate of convergence for Algorithm 3.1. Suppose that Assumption 3.1 holds and that Step 2 of Algorithm 3.1 has generated a sequence $\{x_i\}_{i=0}^n$. Then, in view of (9) and Proposition 3.2,

$$\begin{aligned}
\psi(x_n) - \psi^* &\leq \psi_{p^*Q}(x_n) - \psi^*_{p^*Q} + \frac{\log q}{p^*} \\
&\leq \left(1 - \frac{k}{p^*}\right)^n \left[\psi_{p^*Q}(x_0) - \psi^*_{p^*Q}\right] + \frac{\log q}{p^*} \\
&\leq \left(1 - \frac{k}{p^*}\right)^n \left[\psi(x_0) + \frac{\log q}{p^*} - \psi(x^*_{pQ})\right] + \frac{\log q}{p^*} \\
&\leq \left(1 - \frac{k}{p^*}\right)^n [\psi(x_0) - \psi^*] + \frac{2\log q}{p^*}, \qquad (28)
\end{aligned}$$

where k is as in Proposition 3.2. We examine the rate at which $\psi(x_n) - \psi^*$ vanishes as $n \to \infty$. As is clear from the right-hand side of (28), $\psi(x_n) - \psi^*$ may not vanish if p^* is a constant as $n \to \infty$. Hence, p^* should be large when n is large. Let $e_0 \triangleq \psi(x_0) - \psi^*$ and, for any $n \in \mathbb{N}$ and $p_n \geq 1$, let

$$e_n \triangleq e_0 \left(1 - \frac{k}{p_n}\right)^n + \frac{2\log q}{p_n}. \qquad (29)$$

In view of (28), the quantity e_n is an upper bound on $\psi(x_n) - \psi^*$ when $p^* = p_n$ in Algorithm 3.1.

Before we present the rate of convergence results for Algorithm 3.1, we need the following trivial technical result.

Lemma 3.2. *For $x \in [0, 1/2]$,*

$$-2x \leq \log(1-x) \leq -x. \tag{30}$$

We next state the rate of convergence of Algorithm 3.1, which shows that the rate is no better than sublinear even for an "optimal" choice of p^*.

Theorem 3.2. *Suppose that Assumption 3.1 holds. Let $\{p_n\}_{n=1}^{\infty}$, with $p_n \geq 1$, $n \in \mathbb{N}$, be a sequence of precision parameters and, for any $n \in \mathbb{N}$, let $\{x_i\}_{i=0}^{n} \subset \mathbb{R}^d$ be a sequence generated by Algorithm 3.1 with $p^* = p_n$. Then,*

$$\liminf_{n\to\infty} \frac{\log e_n}{\log n} \geq -1. \tag{31}$$

If $p_n = \zeta n/\log n$ for all $n \in \mathbb{N}$, with $\zeta \in (0, k]$, where k is as in Proposition 3.2, then

$$\lim_{n\to\infty} \frac{\log e_n}{\log n} = -1. \tag{32}$$

Proof. For any $n \in \mathbb{N}$, we see from (29) that

$$\begin{aligned}
\log e_n &= \log\left(\exp\left[\log e_0 + n\log\left(1-\frac{k}{p_n}\right)\right] + \frac{2\log q}{p_n}\right) \\
&\geq \log\left(\max\left\{\exp\left[\log e_0 + n\log\left(1-\frac{k}{p_n}\right)\right], \frac{2\log q}{p_n}\right\}\right) \\
&= \max\left\{\log\left(\exp\left[\log e_0 + n\log\left(1-\frac{k}{p_n}\right)\right]\right), \log\frac{2\log q}{p_n}\right\}.
\end{aligned}$$

Hence, for any $n \in \mathbb{N}$, $n > 1$,

$$\frac{\log e_n}{\log n} \geq \max\left\{\frac{\log e_0}{\log n} + \frac{n\log\left(1-\frac{k}{p_n}\right)}{\log n}, -\frac{\log p_n}{\log n} + \frac{\log 2}{\log n} + \frac{\log\log q}{\log n}\right\}. \tag{33}$$

Let $\epsilon > 0$ be arbitrary. Then, there exists a $n_0 \in \mathbb{N}$ such that $\log\log q/\log n \geq -\epsilon$ for all $n \geq n_0$. If $\log p_n/\log n \leq 1$ and $n \geq \max\{2, n_0\}$, then

$$\frac{\log e_n}{\log n} \geq -\frac{\log p_n}{\log n} + \frac{\log 2}{\log n} + \frac{\log\log q}{\log n} \geq -\frac{\log p_n}{\log n} - \epsilon \geq -1 - \epsilon. \tag{34}$$

Alternatively, suppose that $\log p_n/\log n > 1$. Hence, $n/p_n < 1$, and if $n \geq 2k$, then $k/p_n \in (0, 1/2]$. Based on Lemma 3.2 and (33),

$$\frac{\log e_n}{\log n} \geq \frac{\log e_0}{\log n} + \frac{n\log\left(1-\frac{k}{p_n}\right)}{\log n} \geq \frac{\log e_0}{\log n} + \frac{n\left(-\frac{2k}{p_n}\right)}{\log n} \geq \frac{\log e_0}{\log n} - \frac{2k}{\log n} \tag{35}$$

for all $n \geq 2k$ such that $\log p_n / \log n > 1$. Thus, there exists $n_1 \geq \max\{n_0, 2k\}$ such that

$$\frac{\log e_0}{\log n} - \frac{2k}{\log n} \geq -1 - \epsilon \tag{36}$$

for all $n \geq n_1$. Hence, for all $n \geq n_1$,

$$\frac{\log e_n}{\log n} \geq -1 - \epsilon. \tag{37}$$

Since ϵ is arbitrary, (31) then follows.

Next, we will prove the second part of the theorem. Since $p_n = \zeta n / \log n$, where $\zeta \in (0, k]$, and from (29),

$$\log e_n = \log\left(\exp\left[\log e_0 + n \log\left(1 - \frac{k \log n}{\zeta n}\right)\right] + \frac{2 \log q \log n}{\zeta n}\right). \tag{38}$$

There exists $n_2 \in \mathbb{N}$ such that $k \log n / \zeta n \in [0, 1/2]$ for all $n \geq n_2$. Thus, by Lemma 3.2

$$\begin{aligned} & \log\left(\exp\left[\log e_0 + n\left(-\frac{2k \log n}{\zeta n}\right)\right] + \frac{2 \log q \log n}{\zeta n}\right) \\ \leq\ & \log e_n \\ \leq\ & \log\left(\exp\left[\log e_0 + n\left(-\frac{k \log n}{\zeta n}\right)\right] + \frac{2 \log q \log n}{\zeta n}\right) \end{aligned} \tag{39}$$

for all $n \geq n_2$. We first consider the lower bound in (39),

$$\begin{aligned} & \log\left(\exp\left[\log e_0 + n\left(-\frac{2k \log n}{\zeta n}\right)\right] + \frac{2 \log q \log n}{\zeta n}\right) \\ =\ & \log\left(\frac{2 \log q \log n}{\zeta n}\left[\frac{\exp\left(\log e_0 + \log n^{-2k/\zeta}\right)}{\frac{2 \log q \log n}{\zeta n}} + 1\right]\right) \\ =\ & \log\left(\frac{2 \log q \log n}{\zeta n}\right) + \log\left(\frac{e_0 \zeta n^{1 - \frac{2k}{\zeta}}}{2 \log q \log n} + 1\right). \end{aligned} \tag{40}$$

Since $\zeta \in (0, k]$ and by continuity of the $\log(\cdot)$ function,

$$\lim_{n \to \infty} \log\left(\frac{e_0 \zeta n^{1 - \frac{2k}{\zeta}}}{2 \log q \log n} + 1\right) = 0. \tag{41}$$

Continuing from (40), and using (41), we obtain that

$$\begin{aligned} & \lim_{n \to \infty} \frac{\log\left(\frac{2 \log q \log n}{\zeta n}\right) + \log\left(\frac{e_0 \zeta n^{1 - \frac{2k}{\zeta}}}{2 \log q \log n} + 1\right)}{\log n} \\ =\ & \lim_{n \to \infty} \frac{\log\left(\frac{2 \log q \log n}{\zeta n}\right)}{\log n} + \lim_{n \to \infty} \frac{\log\left(\frac{e_0 \zeta n^{1 - \frac{2k}{\zeta}}}{2 \log q \log n} + 1\right)}{\log n} \\ =\ & \lim_{n \to \infty} \frac{\log 2 + \log\log q + \log\log n - \log \zeta - \log n}{\log n} \\ =\ & -1. \end{aligned} \tag{42}$$

Similar arguments lead to the result that the upper bound in (39) also tends to -1, as $n \to \infty$. Hence, the conclusion follows. □

Theorem 3.2 implies that for large n, e_n is no smaller than approximately $1/n$ for any choice of the precision parameter p_n. Moreover, with the "optimal" choice of $p_n = \zeta n / \log n$, $e_n \approx 1/n$. Hence, the rate of convergence of e_n is sublinear. Since e_n is an upper bound on the distance to the optimal value after n iterations of Algorithm 3.1 with $p^* = p_n$, Algorithm 3.1 has rate of convergence no better than sublinear as the next result formalizes.

Corollary 3.1. *Suppose that Assumption 3.1 holds. Let $\{p_n\}_{n=1}^{\infty}$, with $p_n \geq 1$, $n \in \mathbb{N}$, be a sequence of precision parameters and, for any $n \in \mathbb{N}$, let $\{x_i\}_{i=0}^{n} \subset \mathbb{R}^d$ be a sequence generated by Algorithm 3.1 with $p^* = p_n = \zeta n / \log n$, with $\zeta \in (0, k]$, where k is as in Proposition 3.2. Then*

$$\limsup_{n \to \infty} \frac{\log(\psi(x_n) - \psi^*)}{\log n} \leq -1. \tag{43}$$

Proof. From (28) and (29), $\psi(x_n) - \psi^* \leq e_n$ for all $n \in \mathbb{N}$. Thus, for all $n \in \mathbb{N}, n > 1$,

$$\frac{\log(\psi(x_n) - \psi^*)}{\log n} \leq \frac{\log e_n}{\log n}. \tag{44}$$

The result then follows from Theorem 3.2. □

SQP algorithms for (P) achieve superlinear rate of convergence; see for example [7, 9]. We note, however, that the computational work per iteration for SQP algorithms as discussed above is at least $O((d+q)^3)$. On the other hand, the computational work per iteration of Algorithm 3.1 is $O(qd^2)$ under Assumption 3.2. Hence, there may be classes of problem instances on which smoothing algorithms may perform better than SQP algorithms. The next section gives two novel smoothing algorithms that aim to manage the precision parameter effectively to avoid ill-conditioning.

4 Smoothing Algorithms

We present two smoothing algorithms to solve (P). The first algorithm, Algorithm 4.1 below, is based on Algorithm 3.2 in [13], but uses a much simpler rule for precision adjustment. The second algorithm, Algorithm 4.2 below, adopts a novel line search rule that aims to ensure descent in $\psi(\cdot)$ and, if that is not possible, increases the precision parameter. Previous smoothing algorithms [6, 13] do not check for descent in $\psi(\cdot)$.

We use the following notation. The ϵ-active set, $\epsilon > 0$, is denoted by

$$Q_\epsilon(x) \stackrel{\triangle}{=} \{j \in Q | \psi(x) - f^j(x) \leq \epsilon\}. \tag{45}$$

Similar to Algorithm 3.2 of [13], we compute a search direction using a $d \times d$ matrix $B_{p\Omega}(x)$. We consider two options. When

$$B_{p\Omega}(x) = I, \tag{46}$$

the $d \times d$ identity matrix, the search direction is equivalent to the steepest descent direction. When

$$B_{p\Omega}(x) = \eta_{p\Omega}(x) I + H_{p\Omega}(x), \tag{47}$$

the search direction is a Quasi-Newton direction, where

$$H_{p\Omega}(x) \stackrel{\triangle}{=} p \left(\sum_{j \in \Omega} \mu_p^j(x) \nabla f^j(x) \nabla f^j(x)^T - \left(\sum_{j \in \Omega} \mu_p^j(x) \nabla f^j(x) \right) \left(\sum_{j \in \Omega} \mu_p^j(x) \nabla f^j(x) \right)^T \right), \tag{48}$$

$$\eta_{p\Omega}(x) \stackrel{\triangle}{=} \max\{0, \delta - e_{p\Omega}(x)\}, \tag{49}$$

and $e_{p\Omega}(x)$ is the smallest eigenvalue of $H_{p\Omega}(x)$.

We next present the two algorithms and their proofs of convergence.

Algorithm 4.1.

Data: $x_0 \in \mathbb{R}^d$.

Parameters: $\alpha, \beta \in (0,1), p_0 \geq 1, \omega = 10 \log q / p_0$, function $B_{p\Omega}(\cdot)$ as in (46) or (47), $\epsilon_0 > 0, \xi > 1, \varsigma > 1$.

Step 1. Set $i = 0, j = 0, \Omega_0 = Q_{\epsilon_0}(x_0)$.

Step 2. Compute the search direction $h_{p_i\Omega_i}(x_i)$ by solving the equation

$$B_{p_i\Omega_i}(x_i) h_{p_i\Omega_i}(x_i) = -\nabla \psi_{p_i\Omega_i}(x_i). \tag{50}$$

Step 3. Compute the stepsize $\lambda_i = \beta^{k_i}$, where k_i is the largest integer k such that

$$\psi_{p_i\Omega_i}(x_i + \beta^k h_{p_i\Omega_i}(x_i)) - \psi_{p_i\Omega_i}(x_i) \leq -\alpha \beta^k \| h_{p_i\Omega_i}(x_i) \|^2 \tag{51}$$

and

$$\psi_{p_i\Omega_i}(x_i + \beta^k h_{p_i\Omega_i}(x_i)) - \psi(x_i + \beta^k h_{p_i\Omega_i}(x_i)) \geq -\omega. \tag{52}$$

Step 4. Set

$$x_{i+1} = x_i + \beta^{k_i} h_{p_i\Omega_i}(x_i), \tag{53}$$

$$\Omega_{i+1} = \Omega_i \cup Q_{\epsilon_i}(x_{i+1}). \tag{54}$$

Step 5. Enter Subroutine 4.1, and go to Step 2 when exit Subroutine 4.1. □

Subroutine 4.1. Adaptive Precision-Parameter Adjustment using Optimality Function

If

$$\theta_{p_i\Omega_i}(x_{i+1}) \geq -\epsilon_i, \tag{55}$$

set $x_j^* = x_{i+1}$, set $p_{i+1} = \xi p_i$, set $\epsilon_{i+1} = \epsilon_i/\varsigma$, replace i by $i+1$, replace j by $j+1$, and exit Subroutine 4.1.

Else, set $p_{i+1} = p_i$, set $\epsilon_{i+1} = \epsilon_i$, replace i by $i+1$, and exit Subroutine 4.1. □

Steps 1 to 4 of Algorithm 4.1 is identical to Algorithm 3.2 of [13]. The key difference between the two algorithms is the simplified rule to adjust p_i in Subroutine 4.1. This difference calls for a different proof of convergence as compared to [13], and will be based on consistent approximation. The next result is identical to Lemma 3.1 in [13].

Lemma 4.1. *Suppose that $\{x_i\}_{i=0}^{\infty} \subset \mathbb{R}^d$ is a sequence constructed by Algorithm 4.1. Then, there exists an $i^* \in \mathbb{N}_0$ and a set $\Omega^* \subset Q$ such that working sets $\Omega_i = \Omega^*$ for all $i \geq i^*$.*

Proof. By construction, the cardinality of the working sets $\{\Omega_i\}_{i=0}^{\infty}$ is monotonically increasing. Since the set Q is finite, the lemma must be true. □

Theorem 4.1. *Suppose that Assumption 2.1 holds. Then, any accumulation point $x^* \in \mathbb{R}^d$ of a sequence $\{x_j^*\}_{j=0}^{\infty} \subset \mathbb{R}^d$ constructed by Algorithm 4.1 satisfies the first-order optimality condition $\theta_Q(x^*) = 0$.*

Proof. Let $\Omega^* \subset Q$ and $i^* \in \mathbb{N}_0$ be as in Lemma 4.1, where $\Omega_i = \Omega^*$ for all $i \geq i^*$. As Algorithm 4.1 has the form of Master Algorithm Model 3.3.12 in [17] for all $i \geq i^*$, we conclude based on Theorem 3.3.13 in [17] that any accumulation point x^* of a sequence $\{x_j^*\}_{j=0}^{\infty}$ constructed by Algorithm 4.1 satisfies $\theta_{\Omega^*}(x^*) = 0$. The assumptions required to invoke Theorem 3.3.13 in [17] are (i) continuity of $\psi_{\Omega^*}(\cdot), \psi_{p\Omega^*}(\cdot), \theta_{\Omega^*}(\cdot)$, and $\theta_{p\Omega^*}(\cdot)$, $p > 0$, which follows by Assumption 2.1, Proposition 2.1(i), Theorem 2.1.6 of [17], and Proposition 2.1(iii); (ii) the pair $((P_{p\Omega^*}), \theta_{p\Omega^*}(\cdot))$ must be a consistent approximation to $((P_{\Omega^*}), \theta_{\Omega^*}(\cdot))$, which follows by Theorem 2.1; and (iii) if Steps 1 to 4 of Algorithm 4.1 are applied repeatedly to $(P_{p\Omega^*})$ with a fixed $p > 0$, then every accumulation point $\hat{x}$ of a sequence $\{x_k\}_{k=0}^{\infty}$ constructed must be a stationary point of $(P_{p\Omega^*})$, i.e., $\theta_{p\Omega^*}(\hat{x}) = 0$, which follows by Theorem 3.2 in [13].

Since $\theta_{\Omega^*}(x^*) = 0$, from (14), there exists a $\mu \in \Sigma_{\Omega^*}$ such that

$$\sum_{j\in\Omega^*} \mu^j(\psi_{\Omega^*}(x^*) - f^j(x^*)) + \tfrac{1}{2}\left\|\sum_{j\in\Omega^*} \mu^j \nabla f^j(x^*)\right\|^2 = 0. \tag{56}$$

Let $\pi \in \Sigma_Q$, $\pi^j = 0$ for $j \in Q - \Omega^*$, and $\pi^j = \mu^j$ for $j \in \Omega^*$. Thus, it follows from (14) that

$$\theta_Q(x^*) \geq -\sum_{j\in Q} \pi^j(\psi(x^*) - f^j(x^*)) - \tfrac{1}{2}\left\|\sum_{j\in Q} \pi^j \nabla f^j(x^*)\right\|^2 = 0. \tag{57}$$

Since $\theta_Q(\cdot)$ is a nonpositive function, the result follows. □

Algorithm 4.2.

Data: $x_0 \in \mathbb{R}^d$.

Parameters: $\alpha, \beta \in (0,1)$, function $B_{p\Omega}(\cdot)$ as in (46) or (47), $\epsilon > 0, \delta \geq 1, p_0 \geq 1, \hat{p} \gg p_0, \kappa \gg 1, \xi > 1, \gamma > 0, \nu \in (0,1), \Delta p \geq 1$.

Step 0. Set $i = 0, \Omega_0 = Q_\epsilon(x_0), k_{-1} = 0$.

Step 1. Compute $B_{p_i\Omega_i}(x_i)$ and its largest eigenvalue $\sigma^{\max}_{p_i\Omega_i}(x_i)$. If

$$\sigma^{\max}_{p_i\Omega_i}(x_i) \geq \kappa, \tag{58}$$

compute the search direction

$$h_{p_i\Omega_i}(x_i) = -\nabla\psi_{p_i\Omega_i}(x_i). \tag{59}$$

Else, compute the search direction $h_{p_i\Omega_i}(x_i)$ by solving the equation

$$B_{p_i\Omega_i}(x_i)h_{p_i\Omega_i}(x_i) = -\nabla\psi_{p_i\Omega_i}(x_i). \tag{60}$$

Step 2a. Compute a tentative Armijo stepsize based on working set Ω_i, starting from the eventual stepsize of the previous iterate k_{i-1}, i.e., determine

$$\lambda_{p_i\Omega_i}(x_i) = \max_{l\in\{k_{i-1},k_{i-1}+1,\ldots\}} \{\beta^l | \psi_{p_i\Omega_i}(x_i + \beta^l h_{p_i\Omega_i}(x_i)) - \psi_{p_i\Omega_i}(x_i) \leq \alpha\beta^l \langle \nabla\psi_{p_i\Omega_i}(x_i), h_{p_i\Omega_i}(x_i)\rangle\}. \tag{61}$$

Set

$$y_i = x_i + \beta^l h_{p_i\Omega_i}(x_i). \tag{62}$$

Step 2b. Forward track from y_i along direction $h_{p_i\Omega_i}(x_i)$ as long as $\psi(\cdot)$ continues to decrease using the following subroutine.

Substep 0. Set $l' = l$,

$$z_{il'} = x_i + \beta^{l'} h_{p_i\Omega_i}(x_i) \text{ and } z_{il'-1} = x_i + \beta^{l'-1} h_{p_i\Omega_i}(x_i). \tag{63}$$

Substep 1. If

$$\psi(z_{il'-1}) < \psi(z_{il'}), \tag{64}$$

replace l' by $l'-1$, set $z_{il'-1} = x_i + \beta^{l'-1} h_{p_i\Omega_i}(x_i)$, and repeat Substep 1.

Else, set $z_i = z_{il'}$.

Substep 2. If $p_i \leq \hat{p}$, go to Step 3. Else, go to Step 4.

Step 3. If

$$\psi(z_i) - \psi(x_i) \leq -\frac{\gamma}{{p_i}^{\nu}}, \tag{65}$$

set $x_{i+1} = z_i, p_{i+1} = p_i, k_i = l'$, set $\Omega_{i+1} = \Omega_i \cup Q_\epsilon(x_{i+1})$, replace i by $i+1$, and go to Step 1. Else, replace p_i by ξp_i, replace Ω_i by $\Omega_i \cup Q_\epsilon(z_i)$, and go to Step 1.

Step 4. If

$$\psi(z_i) - \psi(x_i) \leq -\frac{\gamma}{{p_i}^{\nu}}, \tag{66}$$

set $x_{i+1} = z_i, k_i = l'$, set $p_{i+1} = p_i + \Delta p$, set $\Omega_{i+1} = \Omega_i \cup Q_\epsilon(x_{i+1})$, replace i by $i+1$, and go to Step 1.

Else, set $x_{i+1} = y_i, k_i = l$, set $p_{i+1} = p_i + \Delta p$, set $\Omega_{i+1} = \Omega_i \cup Q_\epsilon(x_{i+1})$, replace i by $i+1$, and go to Step 1. □

As is standard in stabilized Newton methods (see for example Section 1.4.4 of [17]), Algorithm 4.2 switches to the steepest descent direction if $B_{p\Omega}(\cdot)$ is given by (47) and the largest eigenvalue of $B_{p\Omega}(\cdot)$ is large; see Step 1. Compared to Algorithm 3.2 in [13], which increases p when the smoothed function gradient is small, Algorithm 4.2 increases the precision parameter only when it does not produce sufficient descent in $\psi(\cdot)$, as verified by (65) and (66). A small precision parameter may produce an ascent direction in $\psi(\cdot)$ due to the poor accuracy of the smoothed function approximation. Thus, insufficient descent is a signal that the precision parameter may be too small. All existing smoothing algorithms only ensure that $\psi_{p\Omega}(\cdot)$ decreases at each iteration, but do not ensure descent in $\psi(\cdot)$. Another change as compared to [6, 13] relates to the line search. All smoothing algorithms are susceptible to ill-conditioning and small stepsizes. To counteract this difficulty, Algorithm 4.2 moves forward along the search direction starting from the Armijo step, and stops when the next step is not a descent step in $\psi(\cdot)$; see Step 2b.

Algorithm 4.2 has two rules for increasing p_i. In the early stages of the calculations, i.e., when $p_i \leq \hat{p}$, if sufficient descent in $\psi(\cdot)$ is achieved when moving from x_i to z_i ((65) satisfied), then Algorithm 4.2 sets the next iterate x_{i+1} to z_i, retain the current value of the

precision parameter as progress is made towards the optimal solution of (P). However, if (65) fails, then there is insufficient descent and the precision parameter or the working set needs to be modified to generate a better search direction in the next iteration. In late stages of the calculations, i.e., $p_i > \hat{p}$, Algorithm 4.2 accepts every new point generated, even those with insufficient descent, and increases the precision parameter with a constant value.

The next lemma is similar to Lemma 4.1.

Lemma 4.2. *Suppose that $\{x_i\}_{i=0}^{\infty} \subset \mathbb{R}^d$ is a sequence constructed by Algorithm 4.2. Then, there exists an $i^* \in \mathbb{N}_0$ and a set $\Omega^* \subset Q$ such that working sets $\Omega_i = \Omega^*$ and $\psi_{\Omega^*}(x_i) = \psi(x_i)$ for all $i \geq i^*$.*

Proof. The first part of the proof follows exactly from the proof for Lemma 4.1. Next, since $\widehat{Q}(x_i) \subset \Omega_i$ for all i; see Steps 3 and 4 of Algorithm 4.2, $\psi_{\Omega^*}(x_i) = \psi(x_i)$ for all $i \geq i^*$. □

Lemma 4.3. *Suppose that Assumption 2.1 holds, and that the sequences $\{x_i\}_{i=0}^{\infty} \subset \mathbb{R}^d$ and $\{p_i\}_{i=0}^{\infty} \subset \mathbb{R}$ are generated by Algorithm 4.2. Then, the following properties hold: (i) the sequence $\{p_i\}_{i=0}^{\infty}$ is monotonically increasing; (ii) if the sequence $\{x_i\}_{i=0}^{\infty}$ has an accumulation point, then $p_i \to \infty$ as $i \to \infty$, and $\sum_{i=0}^{\infty} \frac{1}{p_i} = +\infty$.*

Proof. We follow the framework of the proof for Lemma 3.1 of [6]. (i) The precision parameter is adjusted in Steps 3 and 4 of Algorithm 4.2. In Step 3, if (65) is satisfied, then $p_{i+1} = p_i$; if (65) fails, p_i is replaced by $\xi p_i > p_i$. In Step 4, $p_{i+1} = p_i + \Delta p \geq p_i + 1 > p_i$.

(ii) Suppose that Algorithm 4.2 generates the sequence $\{x_i\}_{i=0}^{\infty}$ with accumulation point $x^* \in \mathbb{R}^d$, but $\{p_i\}_{i=0}^{\infty}$ is bounded from above. The existence of an upper bound on p_i implies that $p_i \leq \hat{p}$ for all $i \in \mathbb{N}_0$, because if not, Algorithm 4.2 will enter Step 4 the first time at some iteration $i' \in \mathbb{N}_0$, and re-enter Step 4 for all $i > i'$, and $p_i \to \infty$ as $i \to \infty$. Thus, the existence of an upper bound on p_i implies that Algorithm 4.2 must never enter Step 4.

The existence of an upper bound on p_i also implies that there exist an iteration $i^* \in \mathbb{N}_0$ such that (65) is satisfied for all $i > i^*$, because if not, p_i will be replaced by ξp_i repeatedly, and $p_i \to \infty$ as $i \to \infty$. This means that $\psi(x_{i+1}) - \psi(x_i) \leq -\gamma/{p_i}^{\nu}$ for all $i > i^*$. Since $p_i \leq \hat{p}$ for all $i \in \mathbb{N}_0$, $\psi(x_i) \to -\infty$ as $i \to \infty$. However, by continuity of $\psi(\cdot)$, and x^* being an accumulation point, $\psi(x_i) \to^K \psi(x^*)$, where $K \subset \mathbb{N}_0$ is some infinite subset. This is a contradiction, so $p_i \to \infty$.

Next, we prove that $\sum_{i=0}^{\infty} \frac{1}{p_i} = +\infty$. Since $p_i \to \infty$, there exist an iteration $i^* \in \mathbb{N}_0$ such that $p_i > \hat{p}$ for all $i \geq i^*$. This means that the precision parameter will be adjusted by the

rule $p_{i+1} = p_i + \Delta p$ for all $i \geq i^*$. The proof is complete by the fact that $\sum_{i=1}^{\infty} 1/i = \infty$. □

Lemma 4.4. *Suppose that Assumption 2.1 holds. Then, for every bounded set $S \subset \mathbb{R}^d$ and parameters $\alpha, \beta \in (0,1)$, there exist a $K_S < \infty$ such that, for all $p \geq 1$, $\Omega \subset Q$, and $x \in S$,*

$$\psi_{p\Omega}(x + \lambda_{p\Omega}(x) h_{p\Omega}(x)) - \psi_{p\Omega}(x) \leq \frac{-\alpha K_S \|\nabla \psi_{p\Omega}(x)\|^2}{p}, \tag{67}$$

where $\lambda_{p\Omega}(x)$ is the stepsize defined by (61), with p_i replaced by p, Ω_i replaced by Ω, and x_i replaced by x.

Proof. If $h_{p\Omega}(x)$ is given by (60) with $B_{p\Omega}(x)$ as in (46), then the result follows by the same arguments as in the proof for Lemma 3.2 of [6]. If $h_{p\Omega}(x)$ is given by (60) with $B_{p\Omega}(x)$ as in (47), then the result follows by similar arguments as in the proof for Lemma 3.4 of [6], but the argument deviates to account for the fact that the lower bound on the eigenvalues of $B_{p\Omega}(x)$ takes on the specific value of 1 in Algorithm 4.2. □

Lemma 4.5. *Suppose that Assumption 2.1 holds and that $\{x_i\}_{i=0}^{\infty} \subset \mathbb{R}^d$ is a bounded sequence generated by Algorithm 4.2. Let $\Omega^* \subset Q$ and $i^* \in \mathbb{N}_0$ be as in Lemma 4.2, where $\Omega_i = \Omega^*$ for all $i \geq i^*$. Then, there exist an accumulation point $x^* \in \mathbb{R}^d$ of the sequence $\{x_i\}_{i=0}^{\infty}$ such that $\theta_{\Omega^*}(x^*) = 0$.*

Proof. Suppose that $\{x_i\}_{i=0}^{\infty}$ is a bounded sequence generated by Algorithm 4.2. Suppose that there exist an $\rho > 0$ such that

$$\liminf_{i \to \infty} \|\nabla \psi_{p_i \Omega^*}(x_i)\| \geq \rho. \tag{68}$$

Since $\{x_i\}_{i=0}^{\infty}$ is a bounded sequence, it has at least one accumulation point. Hence, by Lemma 4.3, $p_i \to \infty$, as $i \to \infty$. Consider two cases, $x_{i+1} = y_i$ or $x_{i+1} = z_i$ in Algorithm 4.2. If $x_{i+1} = y_i$, by Lemma 4.4, there exist an $M < \infty$ such that

$$\psi_{p_i \Omega^*}(x_{i+1}) - \psi_{p_i \Omega^*}(x_i) \leq -\frac{\alpha M \|\nabla \psi_{p_i \Omega^*}(x_i)\|^2}{p_i}, \tag{69}$$

for $i \geq i^*$. Hence,

$$\begin{aligned} \psi_{p_{i+1} \Omega^*}(x_{i+1}) - \psi_{p_i \Omega^*}(x_i) &= \psi_{p_{i+1} \Omega^*}(x_{i+1}) - \psi_{p_i \Omega^*}(x_{i+1}) + \psi_{p_i \Omega^*}(x_{i+1}) - \psi_{p_i \Omega^*}(x_i) \\ &\leq -\frac{\alpha M \|\nabla \psi_{p_i \Omega^*}(x_i)\|^2}{p_i}, \end{aligned} \tag{70}$$

for $i \geq i^*$, where we have used the fact from Proposition 2.1 that

$$\psi_{p_{i+1} \Omega^*}(x_{i+1}) \leq \psi_{p_i \Omega^*}(x_{i+1}), \tag{71}$$

for $i \geq i^*$, because $p_{i+1} \geq p_i$ from Lemma 4.3.

Next, if $x_{i+1} = z_i$, then (65) or (66) is satisfied. It follows from (9) and Lemma 4.2 that,

$$\begin{aligned}
\psi_{p_{i+1}\Omega^*}(x_{i+1}) - \psi_{p_i\Omega^*}(x_i) &\leq \psi_{\Omega^*}(x_{i+1}) + \frac{\log|\Omega^*|}{p_{i+1}} - \psi_{\Omega^*}(x_i) \\
&= \psi(x_{i+1}) + \frac{\log|\Omega^*|}{p_{i+1}} - \psi(x_i) \\
&\leq -\frac{\gamma}{{p_i}^\nu} + \frac{\log|\Omega^*|}{p_i} \\
&= \frac{-\gamma + {p_i}^{\nu-1}\log|\Omega^*|}{{p_i}^\nu}. \qquad (72)
\end{aligned}$$

From (70) and (72), for all $i \geq i^*$,

$$\psi_{p_{i+1}\Omega^*}(x_{i+1}) - \psi_{p_i\Omega^*}(x_i) \leq \max\left\{-\frac{\alpha M\|\nabla\psi_{p_i\Omega^*}(x_i)\|^2}{p_i}, \frac{-\gamma + {p_i}^{\nu-1}\log|\Omega^*|}{{p_i}^\nu}\right\} \qquad (73)$$

By Proposition 2.1, $\|\nabla\psi_{p_i\Omega^*}(x_i)\|$ is bounded because $\{x_i\}_{i=0}^\infty$ is bounded. Since $\nu \in (0,1)$, there exist an $i^{**} \in \mathbb{N}_0$, where $i^{**} \geq i^*$, such that

$$-\frac{\alpha M\|\nabla\psi_{p_i\Omega^*}(x_i)\|^2}{p_i} \geq \frac{-\gamma + {p_i}^{\nu-1}\log|\Omega^*|}{{p_i}^\nu}, \qquad (74)$$

for all $i \geq i^{**}$. Therefore, from (73),

$$\psi_{p_{i+1}\Omega^*}(x_{i+1}) - \psi_{p_i\Omega^*}(x_i) \leq -\frac{\alpha M\|\nabla\psi_{p_i\Omega^*}(x_i)\|^2}{p_i}, \qquad (75)$$

for all $i \geq i^{**}$. Since by Lemma 4.3, $\sum_{i=0}^\infty 1/p_i = +\infty$, it follows from (70) and (75) that

$$\psi_{p_i\Omega^*}(x_i) \to -\infty, \text{ as } i \to \infty. \qquad (76)$$

Let x^* be an accumulation point of $\{x_i\}_{i=0}^\infty$. That is, there exist an infinite subset $K \subset \mathbb{N}_0$ such that $x_i \to^K x^*$. Based on (9), Lemma 4.3, and continuity of $\psi_{\Omega^*}(\cdot)$, it follows that $\psi_{p_i\Omega^*}(x_i) \to^K \psi_{\Omega^*}(x^*)$, as $i \to \infty$, which contradicts (76). Hence, $\liminf_{i\to\infty}\|\nabla\psi_{p_i\Omega^*}(x_i)\| = 0$. Consequently, there exists an infinite subset $K^* \subset \mathbb{N}_0$ and an $x^* \in \mathbb{R}^d$ such that $x_i \to x^*$ and $\theta_{p_i\Omega^*}(x_i) \to^{K^*} 0$, as $i \to \infty$, which implies that $\limsup_{i\to\infty}\theta_{p_i\Omega^*}(x_i) \geq 0$. From Definition 2.1, Theorem 2.1, and the fact that $\theta_{\Omega^*}(\cdot)$ is a nonpositive function, $\theta_{\Omega^*}(x^*) = 0$. □

Theorem 4.2. *Suppose that Assumption 2.1 holds. (i) If Algorithm 4.2 constructs a bounded sequence $\{x_i\}_{i=0}^\infty \subset \mathbb{R}^d$, then there exists an accumulation point $x^* \in \mathbb{R}^d$ of the sequence $\{x_i\}_{i=0}^\infty$ that satisfies $\theta_Q(x^*) = 0$. (ii) If Algorithm 4.2 constructs a finite sequence $\{x_i\}_{i=0}^{i^*} \subset \mathbb{R}^d$, where $i^* < \infty$, then Step 2b constructs an unbounded infinite sequence $\{z_{i^*l'}\}_{l'=l}^{-\infty}$ with*

$$\psi(z_{i^*l'-1}) < \psi(z_{i^*l'}), \qquad (77)$$

for all $l' \in \{l, l-1, l-2, ...\}$, where l is the tentative Armijo stepsize computed in Step 2a.

Proof. First, we consider (i). Let the set $\Omega^* \subset Q$ be as in Lemma 4.2, where $\Omega_i = \Omega^*$ for all $i \geq i^*$. Based on Lemma 4.5, there exist an accumulation point of the sequence $\{x_i\}_{i=0}^{\infty}$, $x^* \in \mathbb{R}^d$ such that $\theta_{\Omega^*}(x^*) = 0$. The conclusion then follows by similar arguments as in Theorem 4.1.

We next consider (ii). Algorithm 4.2 constructs a finite sequence only if it jams in Step 2b. Then, Substep 1 constructs an infinite sequence $\{z_{i^*l'}\}_{l'=l}^{-\infty}$ satisfying (77) for all $l' \in \{l, l-1, l-2, ...\}$. The infinite sequence is unbounded since $h_{p_i\Omega_i}(x_i) \neq 0$ as (77) cannot hold otherwise, and $\beta \in (0, 1)$. □

Next, we consider the run-time complexity of Algorithms 4.1 and 4.2 to achieve a near-optimal solution of (P). Suppose that all functions $f^j(\cdot)$ are active, i.e., $\Omega_i = Q$, near an optimal solution. If $B_{p\Omega}(\cdot)$ is given by (46), then the main computational work in each iteration of Algorithms 4.1 and 4.2 is the calculation of $\nabla\psi_{pQ}(\cdot)$, which takes $O(qd^2)$ operations under Assumption 3.2; see the proof of Theorem 3.1. If $B_{p\Omega}(\cdot)$ is given by (47), then the main computational work is the calculation of (47) and $h_{p\Omega}(x)$. Under Assumption 3.2, it takes $O(qd)$ operations to compute $\mu_p^j(x)$, $j \in Q$, $O(qd^2)$ to compute $\nabla f^j(x)$, $j \in Q$, $O(d^2)$ to multiply $\nabla f^j(x)\nabla f^j(x)^T$, $O(qd^2)$ to sum $\sum_{j\in\Omega} \mu_p^j(x)\nabla f^j(x)\nabla f^j(x)^T$, $O(qd)$ to sum $\sum_{j\in Q} \mu_p^j(x)\nabla f^j(x)$, and $O(d^2)$ to multiply $[\sum_{j\in\Omega} \mu_p^j(x)\nabla f^j(x)][\sum_{j\in\Omega} \mu_p^j(x)\nabla f^j(x)]^T$. The minimum eigenvalue computation of $H_{p\Omega}(x)$ for Algorithm 4.2 takes $O(d^2)$ operations (see [21]). In all, the number of operations to obtain $B_{p\Omega}(x)$ is $O(qd^2)$. A direct method for solving a linear system of equations to compute $h_{p\Omega}(x)$ results in $O(d^3)$ operations; see for example page 63 of [22]. Hence, if $B_{p\Omega}(\cdot)$ is given by (47), then the computational work in each iteration of Algorithms 4.1 and 4.2 is $O(qd^2 + d^3)$.

It is unclear how many iterations Algorithms 4.1 and 4.2 would need to achieve a near-optimal solution as a function of q. However, since they may utilize Quasi-Newton search directions and adaptive precision adjustment, there is reason to believe that the number of iterations will be no larger than that of Algorithm 3.1, which uses the steepest descent direction and a fixed precision parameter. Thus, suppose that for some tolerance $t > 0$, the number of iterations of Algorithms 4.1 and 4.2 to generate $\{x_i\}_{i=0}^{n}$, with the last iterate satisfying $\psi(x_n) - \psi^* \leq t$, is no larger than $O(\log q)$, as is the case for Algorithm 3.1. Then, focusing on q, we find that under these assumptions, the run-time complexity of Algorithms 4.1 and 4.2 to generate a near-optimal solution is no larger than $O(q \log q)$.

5 Numerical Results

We present an empirical comparison of Algorithms 4.1 and 4.2 with algorithms from the literature over a set of problem instances from [6, 7] as well as randomly generated instances; see Appendix A and Table 1. This study appears to be the first systematic comparison of smoothing and SQP algorithms for large-scale problems. We examine problem instances with number of functions up to three orders of magnitude larger than previously reported.

Specifically, we examine (i) Algorithm 2.1 of [7], an SQP algorithm with two QPs that we refer to as SQP-2QP, (ii) Algorithm A in [9], a one-QP SQP algorithm that we refer to as SQP-1QP, (iii) Algorithm 3.2 in [13], a smoothing Quasi-Newton algorithm referred to as SMQN, (iv) Pshenichnyi-Pironneau-Polak min-max algorithm (Algorithm 2.4.1 in [17]), referred to as PPP, (v) an active-set version of PPP as stated in Algorithm 2.4.34 in [17]; see also [23], which we refer to as ϵ-PPP, and (vi) Algorithms 4.1 and 4.2 of the present paper. We refer to Appendix B for details about algorithm parameters. With the exception of PPP and SQP-1QP, the above algorithms incorporate active-set strategies and, hence, appear especially promising for solving large-scale problems. We implement and run all algorithms in MATLAB version 7.7.0 (R2008b) (see [24]) on a 3.73 GHz PC using Windows XP SP3, with 3 GB of RAM. All QPs are solved using TOMLAB CPLEX version 7.0 (R7.0.0) (see [25]) with the Primal Simplex option, which preliminary studies indicate result in the smallest QP run time. We also examined the LSSOL QP solver (see [26]), but its run times appear inferior to that of CPLEX for large-scale QPs arising in the present context.

Algorithm 2.1 of [7] is implemented in the solver CFSQP [27] and we have verified that our MATLAB implementation of that algorithm produces comparable results in terms of number of iterations and run time as CFSQP. We do not directly compare with CFSQP as we find it more valuable to compare different algorithms using the same implementation environment (MATLAB) and the same QP solver (CPLEX).

We carry out a comprehensive study to identify an ϵ (see (45)) in the algorithms' active-set strategies that minimizes the run time for the various algorithms over a wide range of ϵ (1,000 to $1 \cdot 10^{-20}$). We find that SQP-2QP is insensitive to the selection of ϵ, primarily because the algorithm includes additional steps to aggressively trim the working set. ϵ-PPP is highly sensitive to ϵ with variability within a factor of 200 in run times. SMQN, Algorithm 4.1, and Algorithm 4.2 accumulate functions in the working set and therefore are

also sensitive to ϵ. The run times of SMQN, Algorithm 4.1, and Algorithm 4.2 tend to vary within a factor of ten. The below results are obtained using the apparent, best choice of ϵ for each algorithm.

For Algorithm 4.2, we mainly use the Quasi-Newton direction with $B_{p\Omega}(x)$ as defined in (47), because preliminary test runs show that generally, the alternate steepest descent direction with $B_{p\Omega}(x)$ as defined in (46) produces slower run times.

We examine all problem instances from [6, 7] except two that cannot be easily extended to large q. As the problem instances with large dimensionality in [6, 7] do not allow us to adjust the number of functions, we create two additional sets of problem instances. All problem instances are described in detail in Appendix A.

We report run times to achieve a solution x that satisfies

$$\psi(x) - \psi^{\text{target}} \leq t, \tag{78}$$

where ψ^{target} is a target value (see Appendix A) equal to the optimal value (if known) or a slightly adjusted value from the optimal values reported in [6, 7] for smaller q. We use $t = 10^{-5}$. Although this termination criteria is not possible for real-world problems, we find that it is the most useful criterion in this study.

Table 2 summarizes the run times (in seconds) of the various algorithms, with columns 2 and 3 giving the number of variables d and functions q, respectively. Run times in boldface indicate that the particular algorithm has the shortest run time for the specific problem instance. The numerical results in Table 2 indicate that in most problem instances, the run times are shortest for SQP-2QP or Algorithm 4.2. Table 2 indicates that SQP-2QP is significantly more efficient than SQP-1QP for problem instances ProbA-ProbG. This is due to the efficiency of the active-set strategy in SQP-2QP, which is absent in SQP-1QP. However, for ProbJ-ProbM, SQP-1QP is comparable to SQP-2QP. This is because at the optimal solution of ProbJ-ProbM, all the functions are active. This causes the active-set strategy in SQP-2QP to lose its effectiveness as the optimal solution is approached.

Table 2 indicates also that Algorithm 4.1 is significantly more efficient than SMQN for most problem instances. As the only difference between the two algorithms lie in their precision-parameter adjustment scheme, this highlights the sensitivity in the performance of smoothing algorithms to the control of their precision parameters. Table 2 also shows that Algorithm 4.2 is more efficient than Algorithm 4.1 and SMQN for most problem instances.

Table 2 indicates that SQP-2QP is generally more efficient than Algorithm 4.2 for problem instances with small dimensionality, $d \leq 4$ (specifically ProbA-ProbG), and vice versa. This is consistent with the common observation that SQP-type algorithms may be inefficient for instances of large dimensionality; see for example [7].

Table 2 shows that some algorithms return locally optimal solutions for some problem instances (labeled "local" in Table 2). In view of these results, there is an indication that smoothing algorithms (SMQN, Algorithms 4.1 and 4.2) tend to find global minima more frequently than PPP and SQP algorithms.

Table 3 presents similar results as in Table 2, but for larger q. We do not present results for PPP and SQP-1QP as the required QPs exceed the memory limit. The comprehensive sensitivity studies for ϵ show significant improvement for Algorithm 4.2 for ProbJ-ProbM if a large ϵ is used. Hence, we include the results for Algorithm 4.2 with $\epsilon = 1000$ in Table 3. Note that such a large ϵ means that there is effectively no active-set strategy. Sensitivity tests conducted for the other algorithms with a larger ϵ show no improvement in their run times.

The observations from Table 3 are similar to those for Table 2. Table 3 indicates that Algorithm 4.2 with $\epsilon = 1000$ is efficient for ProbJ-ProbM, which are large dimensionality problem instances with a significant number of functions active at the optimal solution. For completeness, the run times for Algorithm 4.2 with $\epsilon = 1000$ for ProbJ-ProbM in Table 2 are 2.8, 14.3, 0.36 and 3.0 seconds respectively, while the run times for the other problem instances are slower than Algorithm 4.2 with $\epsilon = 10^{-20}$.

The results in Tables 2 and 3 indicate that among the algorithms considered, SQP-2QP and Algorithm 4.2 are the most efficient algorithms for minimax problems with a large number of functions. The run times for ProbJ-ProbM indicate that SQP-2QP is less efficient for problem instances with a significant number of the functions that is ϵ-active at the solution, as the active-set strategy loses its effectiveness.

The problem instances from the literature examined in Tables 2 and 3 include either cases with few functions ϵ-active at an optimal solution (ProbA-ProbI) or cases with all functions ϵ-active (ProbJ-ProbM). We also examine randomly-generated problem instances with an intermediate number of functions ϵ-active at the optimal solution; see ProbN in Table 1. The optimal values are unknown in this case but the target values as given in Table 1 appear to be close to the global minima.

Table 4 presents the run times for Algorithm 4.2 and SQP-2QP on ProbN. As the problem instances are relatively well-conditioned, Algorithm 4.2 with $B_{p\Omega}(\cdot)$ given by (46), i.e., a steepest descent (SD) direction, may perform well and is included in the table. The parameter ϵ for Algorithm 4.2 is set to 1000 for this set of problem instances, as preliminary test runs show that it is consistently better than other choices. Table 4 indicates that SQP-2QP is less efficient than Algorithm 4.2 for problem instances with large dimensionality, and where there is a significant number of functions ϵ-active at the optimal solution. The last row in Table 4 shows that for problem instances with high dimensionality ($d \geq 10,000$), the storage of the $d \times d$ $H_{p\Omega}(\cdot)$ matrix for both SQP-2QP and Algorithm 4.2, with $B_{p\Omega}(\cdot)$ given by (47), causes both algorithms to terminate due to memory limitations. Thus, Algorithm 4.2, with $B_{p\Omega}(\cdot)$ given by (46), which do not have any matrix to store, may be a reasonable alternative for problem instances with large dimensionality.

6 Conclusions

This paper focused on finite minimax problems with many functions, which may result from finely discretized semi-infinite minimax or optimal control problems. We conduct run-time complexity and rate of convergence analysis of smoothing algorithms for solving such problems and compare them with those of SQP algorithms. We find that smoothing algorithms may only have the sublinear rate of convergence $1/n$, where n is the number of iterations. However, as shown by the complexity results, their slow rate of convergence may be compensated by small computational work per iteration, which is of order $O(q)$, where q is the number of functions. We present two smoothing algorithms using exponential penalty functions with active-set strategies. The first algorithm is based on a recent smoothing algorithm, but uses a much simpler rule for precision adjustment. The second algorithm implements a novel line search rule that aims to ensure descent in the original objective function, as opposed to descent in the smoothed objective function that existing smoothing algorithms use. We provide a comprehensive numerical comparison between smoothing and SQP algorithms and find that the proposed algorithms are competitive, and especially efficient for large-scale minimax problems with a significant number of functions ϵ-active at stationary points.

Acknowledgments: The second author acknowledges support from AFOSR Young Investigator grant F1ATA08337G003.

References

[1] E. Polak. On the Mathematical Foundations of Nondifferentiable Optimization in Engineering Design. *SIAM Review*, 29:21–89, 1987.

[2] E. Polak, S. Salcudean, and D. Q. Mayne. Adaptive Control of ARMA Plants using Worst Case Design by Semi-Infinite Optimization. *IEEE Transactions on Automatic Control*, 32:388–397, 1987.

[3] X. Cai, K. Teo, X. Yang, and X. Zhou. Portfolio Optimization Under a Minimax Rule. *Management Science*, 46(7):957–972, 2000.

[4] V. F. Demyanov and V. N. Malozemov. *Introduction to Minimax*. Wiley, New York, New York, 1974.

[5] E. R. Panier and A. L. Tits. A Globally Convergent Algorithm with Adaptively Refined Discretization for Semi-Infinite Optimization Problems arising in Engineering Design. *IEEE Transactions on Automatic Control*, 34(8):903–908, 1989.

[6] E. Polak, J. O. Royset, and R. S. Womersley. Algorithms with Adaptive Smoothing for Finite Minimax Problems. *J. Optimization Theory and Applications*, 119(3):459–484, 2003.

[7] J. L. Zhou and A. L. Tits. An SQP Algorithm for Finely Discretized Continuous Minimax Problems and other Minimax Problems with Many Objective Functions. *SIAM J. Optimization*, 6(2):461–487, 1996.

[8] E. Obasanjo, G. Tzallas-Regas, and B. Rustem. An Interior-Point Algorithm for Nonlinear Minimax Problems. *J. Optimization Theory and Applications*, 144:291–318, 2010.

[9] Z. Zhu, X. Cai, and J. Jian. An Improved SQP Algorithm for Solving Minimax Problems. *Applied Mathematics Letters*, 22(4):464–469, 2009.

[10] J. F. Sturm and S. Zhang. A Dual and Interior-Point Approach to Solve Convex Min-Max Problems. In D. Z. Du and P. M. Pardalos, editors, *Minimax and Applications*, pages 69–78. Kluwer Academic Publishers, 1995.

[11] L. Luksan, C. Matonoha, and J. Vlcek. Primal Interior-Point Method for Large Sparse Minimax Optimization. Technical Report 941, Institute of Computer Science, Academy of Sciences of the Czech Republic, Prague, Czech Republic, 2005.

[12] F. Ye and H. Liu and S. Zhou and S. Liu. A Smoothing Trust-Region Newton-CG Method for Minimax Problem. *Applied Mathematics and Computation*, 199(2):581–589, 2008.

[13] E. Polak, R. S. Womersley, and H. X. Yin. An Algorithm Based on Active Sets and Smoothing for Discretized Semi-Infinite Minimax Problems. *J. Optimization Theory and Applications*, 138:311–328, 2008.

[14] X. Li. An Entropy-Based Aggregate Method for Minimax Optimization. *Engineering Optimization*, 18:277–285, 1992.

[15] S. Xu. Smoothing Method for Minimax Problems. *Computational Optimization and Applications*, 20:267–279, 2001.

[16] B. W. Kort and D. P. Bertsekas. A New Penalty Function Algorithm for Constrained Minimization. In *Proceedings 1972 IEEE Conf. Decision and Control*, New Orlean, Louisiana, 1972.

[17] E. Polak. *Optimization. Algorithms and Consistent Approximations.* Springer, New York, New York, 1997.

[18] E. Polak. Smoothing Techniques for the Solution of Finite and Semi-Infinite Min-Max-Min Problems. In G. D. Pillo and A. Murli, editors, *High Performance Algorithms and Software for Nonlinear Optimization.* Kluwer Academic Publishers, Dordrecht, Netherlands, 2003.

[19] H. Urruty and J. Baptiste. *Convex Analysis and Minimization Algorithms 1. Fundamentals.* Springer, Berlin, 1996.

[20] R. D. C. Monteiro and I. Adler. Interior Path Following Primal-Dual Algorithms. Part II: Convex Quadratic Programming. *Mathematical Programming*, 44(1):43–66, 1989.

[21] T. G. Wright and L. N. Trefethen. Large-Scale Computation of Pseudospectra Using ARPACK and Eigs. *SIAM Journal on Scientific Computing*, 23(2):591–605, 2001.

[22] G. Hammerlin and K. H. Hoffmann. *Numerical Mathematics*. Springer, New York, 1991.

[23] E. Polak. On the Convergence of the Pshenichnyi-Pironneau-Polak Minimax Algorithm with an Active Set Strategy. *Journal of Optimization Theory and Applications*, 138(2):305–309, 2008.

[24] Mathworks Inc. *MATLAB 7 Getting Started Guide*. Natick, MA, 2009.

[25] Tomlab Optimization Inc. *User's Guide for TOMLAB/CPLEX v12.1*. Pullman, WA, 2009.

[26] P. E. Gill, S. J. Hammarling, W. Murray, M. A. Saunders, and M. H. Wright. *User's Guide for LSSOL (Version 1.0): a Fortran Package for Constrained Linear Least-Squares and Convex Quadratic Programming*. Stanford, CA, 1986.

[27] C. Lawrence, J. L. Zhou, and A. L. Tits. User's Guide for CFSQP Version 2.5: A C Code for Solving (Large Scale) Constrained Nonlinear (Minimax) Optimization Problems, Generating Iterates Satisfying All Inequality Constraints. Technical report, 1997.

[28] R. A. Brualdi. *Introductory Combinatorics*. Prentice-Hall, Upper Saddle River, NJ, 2004.

[29] Z. Zhu and K. Zhang. A Superlinearly Convergent Sequential Quadratic Programming Algorithm for Minimax Problems. *Chinese Journal of Numerical Mathematics and Applications*, 27(4):15–32, 2005.

Appendix A. Problem Instances

Table 1 describes the problem instances used. Most columns are self-explanatory. Columns 2 and 3 give the number of variables d and functions q, respectively. The target values (column 7) are equal to the optimal values (if known) or a slightly adjusted value from the optimal

values reported in [6, 7] for smaller q. The same target values are used for ProbA-ProbM in Tables 2 and 3.

In this appendix, we denote components of $x \in \mathbb{R}^d$ by subscripts, i.e., $x = (x_1, x_2, ..., x_d) \in \mathbb{R}^d$. When the problem is given in semi-infinite form, as in (80a) - (80i) below, the set Y is discretized into q equally spaced points if

$$\psi(x) = \max_{y \in Y} \phi(x, y), \tag{79a}$$

and $q/2$ equally spaced points if

$$\psi(x) = \max_{y \in Y} |\phi(x, y)|. \tag{79b}$$

ProbA is defined by (79a) and (80a) below, while ProbB-ProbI are defined by (79b) and (80b)-(80i) below, respectively.

$$\phi(x, y) = (2y^2 - 1)x + y(1 - y)(1 - x), \quad Y = [0, 1] \tag{80a}$$

$$\phi(x, y) = (1 - y^2) - (0.5x^2 - 2yx), \quad Y = [-1, 1], \tag{80b}$$

$$\phi(x, y) = y^2 - (yx_1 + x_2 \exp(y)), \quad Y = [0, 2], \tag{80c}$$

$$\phi(x, y) = \frac{1}{1 + y} - x_1 \exp(yx_2), \quad Y = [-0.5, 0.5], \tag{80d}$$

$$\phi(x, y) = \sin y - (y^2 x_3 + yx_2 + x_1), \quad Y = [0, 1], \tag{80e}$$

$$\phi(x, y) = \exp(y) - \frac{x_1 + yx_2}{1 + yx_3}, \quad Y = [0, 1], \tag{80f}$$

$$\phi(x, y) = \sqrt{y} - [x_4 - (y^2 x_1 + yx_2 + x_3)^2], \quad Y = [0.25, 1], \tag{80g}$$

$$\phi(x, y) = \frac{1}{1 + y} - [x_1 \exp(yx_3) + x_2 \exp(yx_4)], \quad Y = [-0.5, 0.5], \tag{80h}$$

$$\phi(x, y) = \frac{1}{1 + y} - [x_1 \exp(yx_4) + x_2 \exp(yx_5) + x_3 \exp(yx_6)], \quad Y = [-0.5, 0.5], \tag{80i}$$

ProbJ-ProbM are defined by $\psi(x) = \max_{j \in Q} f^j(x)$, with $f^j(x)$ as given in (80j)-(80m) below, respectively.

$$f^j(x) = x_j^2, \quad j = \{1, ..., q\}, \tag{80j}$$

$$f^j(x) = x_{(j-1)2+1}^2 + x_{2j}^2, \quad j = \{1, ..., q\}, \tag{80k}$$

$$f^j(x) = x_{(j-1)4+1}^2 + x_{(j-1)4+2}^2 + x_{(j-1)4+3}^2 + x_{4j}^2, \quad j = \{1, ..., q\}, \tag{80l}$$

$$f^j(x) = x_{k_j}^2 + x_{l_j}^2, \quad j = \left\{1, 2, 3, ..., \binom{d}{2}\right\}, \tag{80m}$$

where (k_j, l_j) are all the different 2-combinations (see Section 3.3 of [28]) of $\{1, 2, 3, ..., d\}$, and

$$f^j(x) = a_j x_i^2 + b_j x_i + c_j, j = \{1, ..., q\}, \tag{80n}$$

where $i = \left\lceil \frac{j}{q/d} \right\rceil$, and the constants a_j, b_j, c_j are randomly generated from a uniform distribution on $[0.5, 1]$.

Appendix B. Algorithm Details and Parameters

This appendix provides details on the algorithms implemented.

PPP. Pshenichnyi-Pironneau-Polak min-max algorithm (Algorithm 2.4.1 in [17]) with $\alpha = 0.5, \beta = 0.8$, and $\delta = 1$. We use the same Armijo stepsize rule parameters α and β for all algorithms.

ϵ-PPP. ϵ-Active PPP algorithm (Algorithm 2.4.34 in [17] and the proof of convergence in [23]) with the same parameters as above. The algorithm implemented is the more recent version in [23], which implements the primal form of the optimality function. Preliminary experiments show that the primal form is more efficient for large-scale problems with a large number of functions than the equivalent dual form on page 176 of [17].

SQP-2QP. Sequential Quadratic Programming with two QPs in each iteration; see Algorithm 2.1 of [7]. We use the algorithm parameters recommended in [7] as well as monotone line search. (We examined the use of nonmonotone line search in CFSQP, but find it inferior to monotone line search on the set of problem instances and therefore implemented the latter approach.)

SQP-1QP. Sequential Quadratic Programming with one QP in each iteration; see Algorithm A in [9]. As there are no proposed parameter settings in [9], the algorithm parameters used are the mid-point values stated in Algorithm A, $\alpha = 0.25$ (α in this algorithm is not the Armijo parameter), $\tau = 2.5$, and matrix H_0 = identity matrix. The same parameter settings for α and H_0 are used by a co-author in a similar algorithm to solve the minimax problem; see [29].

SMQN. Smoothing Quasi-Newton algorithm; see Algorithm 3.2 in [13]. There are no proposed parameter settings in [13]. We adopt commonly-used parameters from other smoothing algorithms, $p_0 = 1$ and $B(\cdot)$ = Identity matrix. For the Penalty-Parameter

Adjustment subroutine, which is the same as that in [6], we use Case (A) of [6], which is shown to be comparable to Case (B).

Algorithm 4.1. The algorithm parameters used are the same as for SMQN, except for the parameters in the different Adaptive Penalty Parameter Adjustment subroutine, $\xi = 2, \varsigma = 2$.

Algorithm 4.2. The algorithm parameters used are $t = 10^{-5}, p_0 = 1, \hat{p} = (\log q/t) \cdot 10^{10}, \kappa = 10^{30}, \alpha = 0.5, \beta = 0.8, \xi = 2, \gamma = t \cdot 10^{-10}, \nu = 0.5, \Delta p = 10$.

Table 1: Problem instances. An asterisk * indicates that the problem instance is created by the authors.

Instance	d	q	$\psi(x)$	Convexity	Initial point	Target value	Ref.
ProbA	1	q	(79a), (80a)	Convex	5	0.1783942	[6]
ProbB	1	q	(79b), (80b)	Non-convex	1	1.0000100	[7]
ProbC	2	q	(79b), (80c)	Convex	$(1, 1)$	0.5382431	[7]
ProbD	2	q	(79b), (80d)	Non-convex	$(1, -1)$	0.0871534	[7]
ProbE	3	q	(79b), (80e)	Convex	$(1, 1, 1)$	0.0045048	[6]
ProbF	3	q	(79b), (80f)	Non-convex	$(1, 1, 1)$	0.0042946	[7]
ProbG	4	q	(79b), (80g)	Non-convex	$(1, 1, 1, 1)$	0.0026500	[6]
ProbH	4	q	(79b), (80h)	Non-convex	$(1, 1, -3, -1)$	0.0020688	[7]
ProbI	6	q	(79b), (80i)	Non-convex	$(1, 1, 1, -7, -3, -1)$	0.0006242	[7]
ProbJ	q	q	(2), (80j)	Convex	$(\frac{2}{q}, \frac{4}{q}, \frac{6}{q}, ..., 1, -1 - \frac{2}{q}, ..., -2)$	0	[6]
ProbK	$2q$	q	(2), (80k)	Convex	$(\frac{1}{q}, \frac{2}{q}, \frac{3}{q}, ..., 1, -1 - \frac{1}{q}, ..., -2)$	0	[6]
ProbL	$4q$	q	(2), (80l)	Convex	$(\frac{1}{2q}, \frac{2}{2q}, \frac{3}{2q}, ..., 1, -1 - \frac{1}{2q}, ..., -2)$	0	[6]
ProbM	d	q	(2), (80m)	Convex	$(\frac{2}{d}, \frac{4}{d}, \frac{6}{d}, ..., 1, -1 - \frac{2}{d}, ..., -2)$	0	*
ProbN(i)	10	10,000	(2), (80n)	Convex	$(\frac{2}{d}, \frac{4}{d}, \frac{6}{d}, ..., 1, -1 - \frac{2}{d}, ..., -2)$	0.9278640	*
ProbN(ii)	100	10,000	(2), (80n)	Convex	$(\frac{2}{d}, \frac{4}{d}, \frac{6}{d}, ..., 1, -1 - \frac{2}{d}, ..., -2)$	0.9313887	*
ProbN(iii)	1,000	10,000	(2), (80n)	Convex	$(\frac{2}{d}, \frac{4}{d}, \frac{6}{d}, ..., 1, -1 - \frac{2}{d}, ..., -2)$	0.9288089	*
ProbN(iv)	10	100,000	(2), (80n)	Convex	$(\frac{2}{d}, \frac{4}{d}, \frac{6}{d}, ..., 1, -1 - \frac{2}{d}, ..., -2)$	0.9307828	*
ProbN(v)	100	100,000	(2), (80n)	Convex	$(\frac{2}{d}, \frac{4}{d}, \frac{6}{d}, ..., 1, -1 - \frac{2}{d}, ..., -2)$	0.9340950	*
ProbN(vi)	1,000	100,000	(2), (80n)	Convex	$(\frac{2}{d}, \frac{4}{d}, \frac{6}{d}, ..., 1, -1 - \frac{2}{d}, ..., -2)$	0.9366594	*
ProbN(vii)	1,000	1,000,000	(2), (80n)	Convex	$(\frac{2}{d}, \frac{4}{d}, \frac{6}{d}, ..., 1, -1 - \frac{2}{d}, ..., -2)$	0.9358776	*
ProbN(viii)	1,000	10,000,000	(2), (80n)	Convex	$(\frac{2}{d}, \frac{4}{d}, \frac{6}{d}, ..., 1, -1 - \frac{2}{d}, ..., -2)$	0.9369501	*
ProbN(ix)	10,000	100,000	(2), (80n)	Convex	$(\frac{2}{d}, \frac{4}{d}, \frac{6}{d}, ..., 1, -1 - \frac{2}{d}, ..., -2)$	0.9335266	*

Table 2: Run times (in seconds) for various algorithms. The word "local" means that the algorithm converges to a locally optimal solution that does not satisfy (78), which may occur for non-convex problems. An asterisk * indicates that the algorithm does not satisfy (78) after 6 hours, and $\psi(x) - \psi^{\text{target}} > 10^{-4}$ at termination, while ** indicates $\psi(x) - \psi^{\text{target}} > 10^{-3}$ at termination.

Instance	d	q	PPP	ϵ-PPP ($\epsilon = 10^{-3}$)	SQP-2QP ($\epsilon = 1$)	SQP-1QP	SMQN ($\epsilon = 10^{-20}$)	Algo 4.1 ($\epsilon = 10^{-20}$)	Algo 4.2 ($\epsilon = 10^{-20}$)
ProbA	1	100,000	17.3	2.5	0.45	13.7	0.64	0.41	**0.31**
ProbB	1	100,000	2.5	0.69	**0.06**	131.1	0.31	0.70	0.45
ProbC	2	100,000	15.3	5.0	**0.67**	11.9	12.6	1.9	1.3
ProbD	2	100,000	5.0	7.4	**0.21**	9.9	7.2	1.7	1.5
ProbE	3	100,000	19.5	14.5	**0.59**	18.3	8.1	2.2	2.0
ProbF	3	100,000	28.7	18.8	2.3	24.4	18.2	2.9	**2.1**
ProbG	4	100,000	local	local	**2.4**	79.1	25.3	28.7	20.1
ProbH	4	100,000	211.8	968.4	local	128.9	36.2	31.5	**23.5**
ProbI	6	100,000	37.7	local	local	**31.7**	425.2	512.9	local
ProbJ	1,000	1,000	*	*	1458	1161	**	2212	**465.6**
ProbK	2,000	1,000	*	*	**	8404	**	10620	**2265**
ProbL	400	100	**3.6**	79.7	15.1	14.5	112.0	17.5	4.6
ProbM	100	4950	7.0	160.3	2.7	3.6	4.5	3.6	**2.3**

Table 3: Similar results as in Table 2, but with larger q. The word "local" means that the algorithm converges to a locally optimal solution that does not satisfy (78), which may occur for non-convex problems. An asterisk * indicates that the algorithm does not satisfy (78) after 6 hours, and $\psi(x) - \psi^{\text{target}} > 10^{-4}$ at termination, while ** indicates $\psi(x) - \psi^{\text{target}} > 0.01$ at termination.

Instance	d	q	ϵ-PPP ($\epsilon = 10^{-3}$)	SQP-2QP ($\epsilon = 1$)	SMQN ($\epsilon = 10^{-20}$)	Algo 4.1 ($\epsilon = 10^{-20}$)	Algo 4.2 ($\epsilon = 1000$)	Algo 4.2 ($\epsilon = 10^{-20}$)
ProbA	1	1,000,000	22.5	4.6	4.4	**2.7**	8.6	3.1
ProbB	1	1,000,000	6.1	**0.61**	2.7	4.8	2.5	3.0
ProbC	2	1,000,000	59.4	**7.2**	131.0	15.0	61.9	13.1
ProbD	2	1,000,000	79.3	**2.2**	75.3	12.3	47.5	13.4
ProbE	3	1,000,000	245.0	**5.5**	93.0	12.1	74.5	17.5
ProbF	3	1,000,000	332.9	22.9	185.9	21.7	74.1	**18.1**
ProbG	4	1,000,000	local	**27.2**	257.8	220.1	12227	169.5
ProbH	4	1,000,000	12322	local	362.8	240.4	4157	**238.4**
ProbI	6	1,000,000	local	local	4262	4016	**3717**	local
ProbJ	4,000	4,000	**	**	**	**	**92.8**	**
ProbK	4,000	2,000	**	**	**	**	**91.8**	**
ProbL	4,000	1,000	*	**	**	**	**106.6**	13273
ProbM	200	19900	*	24.7	66.1	917.3	8.6	**2.5**

Table 4: Run times (in seconds) of algorithms on problem instance ProbN. "SD" and "QN" indicate that Algorithm 4.2 uses $B_{p\Omega}(\cdot)$ given by (46) and (47), respectively. The word "mem" indicates that the algorithm terminates due to insufficient memory.

d	q	SQP-2QP ($\epsilon = 1$)	Algo 4.2 SD ($\epsilon = 1000$)	Algo 4.2 QN ($\epsilon = 1000$)
10	10,000	**0.42**	0.64	0.62
100	10,000	0.82	**0.48**	0.54
1,000	10,000	124.9	**0.38**	4.8
10	100,000	4.1	**3.8**	4.2
100	100,000	11.5	**3.8**	4.1
1,000	100,000	mem	**4.3**	9.7
1,000	1,000,000	mem	**37.2**	42.5
1,000	10,000,000	mem	**421.8**	492.5
10,000	100,000	mem	**6.3**	mem